MagView++: Data Exfiltration via CPU Magnetic Signals Under Video Decoding

Xiaoyu Ji, *Member, IEEE*, Juchuan Zhang, *Member, IEEE*, Shan Zou, *Member, IEEE*, Yichao Chen, *Member, IEEE*, Gang Qu, *Fellow, IEEE*, and Wenyuan Xu, *Member, IEEE*

***Abstract*—Air-gapped networks achieve security by using physical isolation to keep the computers and network from the Internet. However, magnetic covert channels based on CPU utilization have been proposed to help secret data to exfiltrate from the Faraday-cage and the air gap. Despite the success of such covert channels, they suffer from the high risk of being detected by the transmitter computer and the challenge of installing malware into such a computer. In this article, we propose `MagView++`, where sensitive information is embedded in other data such as video and can be transmitted over the internal network. When any computer uses the data such as playing the video, the sensitive information will leak through the magnetic signals. The "separation" of information embedding and leaking, combined with the fact that the data can be exfiltrated from any computer in a distributed manner, overcomes these limitations. We demonstrate that CPU utilization for video decoding can be effectively controlled by changing the video frame type, reducing the quantization parameter, and changing the timestamp of the frame, without video quality degradation. We prototype `MagView++` and achieve 8.9 bps throughput with 0.0057 BER when using a smartphone as the receiver, and 59 bps throughput with 0.0025 BER when using a dedicated devices with high sampling rate as the receiver. Experiments under various environments are conducted to show the robustness of `MagView++`. Limitations and possible countermeasures are also discussed.**

***Index Terms*—Covert channel, CPU magnetic field, video codec.**

I. Introduction

AIR-GAPPED networks are those private networks where the computers and other equipment are physically isolated without connection to outside public networks such as the Internet. In addition to the communication on the private internal network [1], some air-gapped networks forbid the use of Wi-Fi, Bluetooth, and infrared [2] as well as the use of memory cards [3], [4] to prevent data leakage. Thus, we have seen many security-aware organizations such as NSA and US Defense Intelligence Agency use air-gapped networks as the infrastructure for their daily operations [5]. However, they are not immune to breaches of covert channels, i.e., channels that are not intended for information transfer but may leak sensitive data, even with low signal-to-noise-ratio (SNR). Common media of a covert channel can be acoustic, ultrasonic, electromagnetic, thermal, or optical [6]. However, with security enhancement, more and more existing covert channels like optical channels, acoustic channels, etc., are being cut off [7].

Low-frequency magnetic field, which is generated by the electric current in CPU modules, is a state-of-the-art covert channel as it can pass the Faraday-Cage and is difficult to detect. By regulating the CPU utilization, sensitive data is encoded into the changes of magnetic strength. Receivers such as magnetometers [8], smartphones [9] can receive and decode the magnetic signal to extract the leaked data. As CPU is an essential part of any computer, the covert channel can be implemented on desktop PCs, servers, laptops, and even embedded systems.

Currently proposed magnetic-field-based covert channels (hereafter we name it magnetic covert channel) [8], [9], however, have two major limitations. First, they require direct regulation of the computer's CPU utilization to embed sensitive information, which can easily attract attention and be caught. Second, malware has to be implanted on the same computer for CPU utilization control and sensitive data exfiltration, which further limits its usage.

In this article, we seek to enhance the practicability and stealthiness of the magnetic covert channel by (1) getting rid of implanting malware on the very computer that is leaking sensitive data and (2) hiding direct CPU utilization regulation. To this end, we need physically decouple the embedding and leaking of sensitive information in order to implant the malware only where sensitive information is embedded; find a "carrier" which contains the embedded sensitive information to control CPU utilization in a stealthy way during leaking.

We observe that video interfaces are ubiquitous in security-aware organizations, including videos from surveillance cameras and promotional videos, etc. Because video playing needs a decoding step, it can be a good candidate for CPU intensive operations. This leads us to the idea of using video encoding to embed information and decoding to manipulate CPU utilization for information leakage. By doing this, the two requirements mentioned above can be satisfied. First, information embedding

Manuscript received 3 January 2022; revised 22 February 2023; accepted 16 March 2023. Date of publication 28 March 2023; date of current version 5 February 2024. This work was supported by the China NSFC under Grants 62222114, 61925109, and 62071428. Recommended for acceptance by A. S. Uluagac. *(Corresponding author: Wenyuan Xu.)*

We follow the local regulations to protect the rights of human participants despite the absence of the Institutional Review Board (IRB).

Xiaoyu Ji, Juchuan Zhang, and Shan Zou are with the College of Electrical Engineering, Zhejiang University, Hangzhou, Zhejiang 310027, China (e-mail: xji@zju.edu.cn; juchuanzhang@zju.edu.cn; 22010094@zju.edu.cn).

Yichao Chen is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yichao@sjtu.edu.cn).

Gang Qu is with the ECE and ISR, University of Maryland, College Park, MD 20742 USA (e-mail: gangqu@umd.edu).

Wenyuan Xu is with the Department of Electronic Engineering, Zhejiang University, Hangzhou, Zhejiang 310027, China (e-mail: xuwenyuan@zju.edu.cn).

Digital Object Identifier 10.1109/TMC.2023.3262400

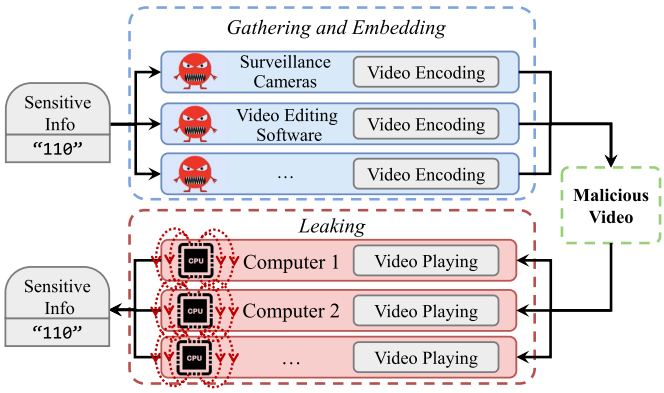


Fig. 1. The concept of separation in MagView++. Top: sensitive information is embedded during video encoding. Bottom: sensitive information distributedly leaks through the magnetic covert channel created whenever a video is played. "Malicious" video with sensitive information can be transmitted internally in the air-gapped network.

and leaking can be separated. Specifically, sensitive information embedding can be done by surveillance cameras or computers having video editing software if malware is implanted on those devices. Therefore, the data exfiltration progress can be distributed, i.e., any devices playing the videos with embedded information will exfiltrate the data, as shown in Fig. 1. Second, since the CPU utilization regulation is covered by the video playing task, the magnetic covert channel becomes stealthy and difficult for human to notice. Fig. 1 depicts the overview of MagView++. Sensitive information gathered by the malicious surveillance camera, for example, can be embedded into malicious videos which can be played by any devices in the internal networks. A smartphone or a dedicated device with a magnetic sensor placed next to the computer playing the video can pick up the magnetic signals and recover the sensitive data.

The design of MagView++ encounters several challenges. First, the re-encoded videos cannot be suspicious visually, i.e., the video content and quality such as resolution cannot be changed. Second, how to keep a high SNR for the magnetic covert channel with background application running on the devices is also a challenge. Third, how to maximize the transmission rate for receivers with different sampling rates should be concerned. To cope with the above challenges, we propose MagView++ with two transmission schemes for smartphone receivers and dedicated receivers respectively. For smartphone receivers, we carefully investigate H.264/AVC [10], a common video encoding standard, and find out that the frame type and the quantization parameter (QP) can control the size of a video frame, and thereby can affect the CPU utilization when decoding video frames. Such a strategy to increase CPU utilization is also validated on H.265 [11]. We also design the ASK modulation, DSSS-like encoding scheme, and use Forward Error Correction (FEC) to increase the robustness of the covert channel. For dedicated receivers whose sampling rates are much higher than those of smartphone receivers, we investigate the variable frame rate (VFR) video, which is achieved by assigning a timestamp to each video frame. We find out that changing the timestamp of a video frame can control the moment when high CPU utilization occurs. We then use Pulse Position Modulation to encode data on the timestamp changes and also use FEC to reduce its bit error rate.

In summary, we have made the following contributions:

- We propose MagView++, a CPU magnetic covert channel hidden in video playing tasks, featuring the separation of data embedding and leaking, to improve the concealment of data exfiltrating.
- We propose a Frame-type-and-QP-based encoding scheme for smartphone receivers and a Timestamp-based encoding scheme for dedicated receivers respectively.
- We prototype MagView++ and achieve 8.9 bps throughput for smartphone receivers and 59 bps throughput for dedicated receivers. Comprehensive experiments are performed on 13 different computers as transmitters and 7 different smart devices and a DAQ as receivers, which shows the robustness of MagView++.

II. Background

In this section, we first introduce the background knowledge of video encoding and decoding for the design to change CPU utilization. Then we provide the principle of how a CPU module can generate magnetic signals and the relationship between CPU utilization and magnetic signal strength.

A. Video Encoding and Decoding

A video is composed of a sequence of frames, i.e., I frame, P frame, and B frame in the H.264/AVC standard [12] and each frame can be viewed as a still image. I frame is encoded without reference, while P frame and B frame are encoded as the differences from a reference frame with motion prediction to reduce video size. Consequently, the size of I frame is larger than the other two.

To reduce the video size, compression is always performed on video frames, by going through the processes of discrete cosine transform (DCT), quantization, and entropy encoding. The DCT step is similar to that in image compression, which is used to reduce the special redundancy of an image. The quantization step is to map the DCT coefficients to a reduced range of values and thus it should be possible to represent the DCT coefficients with fewer bits [12]. Finally, the entropy encoding step is to reduce the redundancy between the compressed data symbols using variable length coding techniques [13]. Among the steps, only the quantization step introduces signal loss and its parameter, i.e., quantization parameter (QP) directly determines the compression performance. Roughly speaking, a smaller QP leads to less efficient compression, a higher bit rate (larger video size), and vice versa. QP value can be configured dynamically per frame.

Video decoding and video rendering are two essential steps for playing a video. Video decoding has exactly the opposite process of video encoding and is implemented by video players, which decode videos in the unit of frame. In general, a video player decodes video frames and puts them into a buffer. Then the video player renders the frame in the buffer. Once a frame in the buffer is rendered, the video player will decode a new frame.

Remarks: According to the changes of frame types (I/P/B) and configurations of QP, the video decoding process should possess a dynamic pattern in terms of computation overhead, which is reflected by CPU utilization. In the meanwhile, since the moment of video frame decoding calculation is determined by the moment of video frame playing in the queue, the CPU utilization pattern can also be controlled by frame timestamps.

B. CPU Module and Magnetic signals

The dynamic power consumption during CPU execution can be estimated as [14]

$$P = a_t C_L V_{dd}^2 f_{clock} \tag{1}$$

where $a_t C_L$ is the effective capacitance being switched to perform a computation, which is related to CPU utilization and the performed specific computation; V_{dd} is the operating voltage of the CPU and f_{clock} is the clock frequency, both of which are variable according to the dynamic frequency scaling (DFS) for energy saving [14]. The DFS policy decreases V_{dd} and f_{clock} with low CPU load and vice versa. Therefore, the power differences between busy and idle states of CPU are determined by both effective switched capacitance and the voltage and frequency scaling caused by DFS. In other words, when the CPU is busy, i.e., the CPU utilization is high, it gains more power consumption than when the CPU is idle.

The total CPU module can be seen as a magnetic dipole. For the sake of simplicity, the magnetic field generated by the CPU module can be represented as $B \propto \frac{I}{r^3}$, where B is the magnetic induction intensity, I is the total current in the CPU module, and r is the distance to the CPU module. Combining the above equations with $P = V_{dd} I$, we have

$$B \propto \frac{a_t C_L V_{dd} f_{clock}}{r^3} \tag{2}$$

As $a_t C_L$, V_{dd} and f_{clock} are all positively correlated with CPU utilization, we conclude that the magnetic induction intensity of the CPU magnetic field is strongly correlated with CPU utilization.

III. Threat Model and Overview

A. Threat Model and Assumptions

The aim of `MagView++` is to use video as a medium for covert communication, e.g, exfiltrating sensitive data from an air-gapped network. `MagView++` compounds of 2 phases: 1) data collecting and embedding; 2) video playing and data receiving. For each phase, we make the necessary assumptions.

1) Data Collecting and Embedding: In this phase, the attacker needs to plant the malware of `MagView++` into the camera firmware or video encoding software for sensitive data collecting and embedding. Since cameras and video encoding software are usually provided by third parties, there is a possibility of supply chain attacks. The assumptions on the malware infection phase are similar to those reported in the literature [8], [9], where the attacker's goal is to exfiltrate sensitive data from air-gapped networks. More details on this that can be done is out of the scope of this article and can be found in [15], [16].

The malware of `MagView++` collects the sensitive data to be transferred. For example, a malicious camera can collect network configuration information such as MAC address, or information from the smart cameras' own statistics, such as foot traffic, etc, while video coding software can collect files on the device, as video encoding software itself requires read and write access on storage space.

`MagView++` embeds data into the video by modifying the video encoding parameters, instead of re-encoding the video. As `MagView++` only changes the parameters to encode video frames (frame type, QP, and timestamp), it is not necessary to change the hardware-based video encoding module of a surveillance camera and thus the real-time performance is guaranteed.

2) Video Playing and Data Receiving: We assume that only one video is being played on a computer, which is reasonable when watching surveillance video replays and promos. We assume that the video is played by third-party video players instead of the players that come with the system, such as the Movie & TV player that comes with Windows 10. This is reasonable as the Movie & TV player does not support decoding H.265 (HEVC) videos, which are widely used in surveillance cameras and are supported by `MagView++`. In order to ensure compatibility, third-party players usually use software decoding by default, so that both the Frame-type-and-QP-based encoding scheme and the Timestamp-based encoding scheme of `MagView++` can successfully embed and transmit data. When using hardware decoding, the Frame-type-and-QP-based encoding scheme fails but the Timestamp-based encoding scheme can still transmit data on some video players such as GOM Player, Pot Player, etc.

Whenever the video is played, the sensitive information leaks through magnetic signals. The attacker can get close to computers on which the malicious video is played and put her smartphone or a small device in disguise close to the chassis of a computer or on the laptop's keyboard to receive the sensitive data. For example, a guest can get into the reception room where a promotional or demonstration video is played or an internal staff can enter the monitoring center of a security organization. Particularly, the attacker can use a very small device, such as a smartwatch, to get close to the video playback device to receive the leaked information and thus avoid being suspected. Another case is that the attacker could plant the malware into an insider's phone and use the insider's phone to collect magnetic signals which is the same as [9]. In this case, no additional permissions are required.

B. `MagView++` Overview

Similar to existing magnetic covert channels [8], [9], `MagView++` is based on the magnetic field during CPU execution. The novelty of `MagView++` is how it manipulates CPU utilization. As we mentioned earlier, the malware will create a "malicious" video which when being played, have specific CPU requests and impact the magnetic field so the receiver can extract the sensitive data from the covert channel.

Aiming at different receivers, we propose `MagView++` for smartphone receivers and dedicated receivers with two different data encoding schemes as shown in Table I. For smartphone

TABLE I
OVERVIEW OF MagView++ FOR SMARTPHONE RECEIVERS AND DEDICATED RECEIVERS

Receiver	Smartphone Receivers	Dedicated Receivers
Sampling rate	$\leq 100Hz$	$\geq 10kHz$
Scheme	Frame-type-and-QP-based	Timestamp-based
Bit rate	8.9 bps	59 bps
Invisibility	Completely invisible	Slight frame loss
Overhead	Video size	None

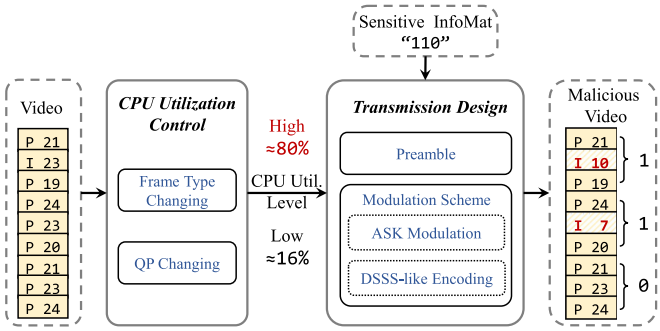


Fig. 2. Technical block diagram of the Frame-type-QP-based scheme for smartphone receivers. To generate a malicious video, Frame-type-and-QP-based Scheme first determines and changes the frame type and QP of each frame of the original video to achieve two different CPU utilization levels, i.e., "High (80%)" and "Low (16%)" and then use the two levels to embed and modulate the sensitive information into video frames. (B frames are omitted for simplicity but are also applicable.)

receivers, the magnetic signals are collected through the smartphone's built-in magnetometer, of which the sampling rate is usually lower than 100 Hz. On the contrary, for dedicated receivers, the sampling rate can be 10 kHz or higher. Below we present an overview of MagView++ for the two cases.

1) MagView++ *for Smartphone Receivers.:* In this case, MagView++ is designed for using a smartphone as a receiver. Since the sampling rate is usually below 100 Hz, it is difficult to obtain the detailed pattern of each decoded frame in the received signal. Based on this condition, we seek to find a way to encode data at higher or lower CPU utilization for several frames of video. Therefore, we propose a *Frame-type-and-QP-based scheme* to embed sensitive data into a video.

Fig. 2 illustrates how sensitive data is embedded into the video to create the "malicious" video. In the CPU utilization control step, both frame type changing and QP changing are used to achieve two different CPU utilization levels. Then in the transmission step, ASK modulation and DSSS-like encoding with preamble are used to modulate the sensitive data on video frames with the two CPU utilization levels. The malicious video is then delivered to computers or laptops (here we call them transmitters) which will play the video. When the video plays on any computer, the sensitive data leaks from the CPU magnetic field and can be picked up by a device with a magnetometer. We present the design and evaluation of MagView++ for smartphone receivers in Sections IV and V respectively.

2) MagView++ *for Dedicated Receivers.:* In this case, MagView++ is designed for using a dedicated device with a higher than 10 kHz magnetic signal sampling rate as a receiver.

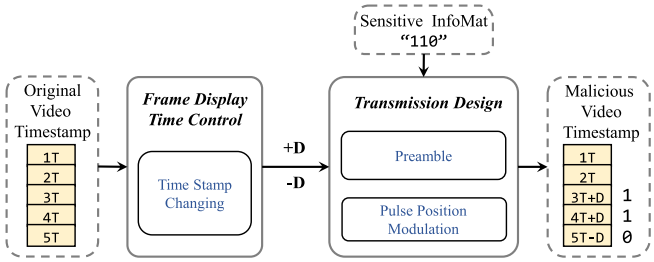


Fig. 3. Technical block diagram of the Timestamp-based scheme for dedicated receivers. To generate a malicious video, MagView++ first changes the timestamp of each frame of the original video to achieve two different timestamp offset levels, i.e., "$nT + D$" and "$nT - D$" and then uses the two offset levels to embed and modulate the sensitive information into video frames.

Due to the high sampling rate, the detailed pattern of each decoded frame in the received signal can be obtained. In other words, we can distinguish the moment when the decoding of each video frame starts. Hence, there is an opportunity to encode data by slightly changing the timestamp of each video frame. Therefore, we propose a *Timestamp-based scheme* to embed sensitive data into a video.

In the Timestamp-based scheme, the timestamp of each frame is used to embed sensitive data as illustrated in Fig. 3. In the frame display time control step, the timestamp of a video frame can be changed by $+D$ or $-D$ to achieve two timestamp offset levels. Then in the transmission step, we use Pulse Position Modulation to modulate the sensitive data on video frames with the two timestamp offset levels. The remaining steps are the same with the Frame-type-and-QP-based scheme. We present the design and evaluation of MagView++ for dedicated receivers in Sections VI and VII respectively.

IV. DESIGN OF MagView++ FOR SMARTPHONE RECEIVERS

In Section II, we conclude that both frame type and QP determine the bit rate on the granularity of the frame. Therefore, it is possible to change the CPU utilization of video decoding by changing frame type and QP. Although frame rate and resolution can also affect the bit rate, they are not supported to be configured and changed sometimes. As a result, we resort to both frame type and QP and incorporate them into a systematic approach to quantitatively output a target CPU utilization.

A. Changing CPU Utilization

1) Changing the Frame Type: As is discussed in Section II, the size of I frames is larger than P and B frames. Therefore, I frames are avoided and P, B frames are preferred by default during the encoding process, unless necessary. As a result, the number of I frames is relatively smaller than that of P, B frames. This provides us the chance to modify a P/B frame to an I frame to gain higher CPU utilization.

2) Changing the Quantization Parameter (QP): It is mentioned in Section II that smaller QP leads to less signal loss and higher bit rate, and thus the CPU utilization during frame decoding increases. With the requirement to keep the original video quality, QP has to be less than a specific value that is

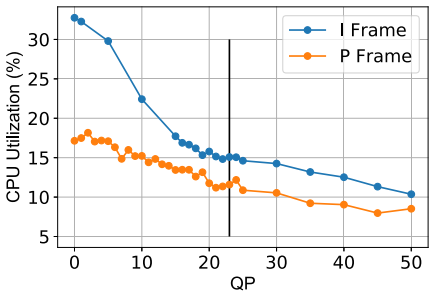


Fig. 4. CPU utilization versus QP.

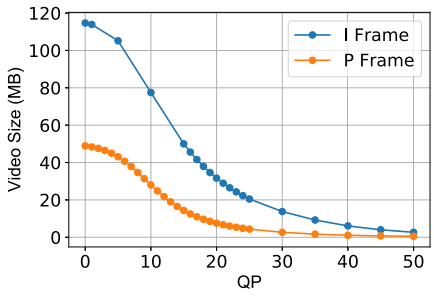


Fig. 5. Video size versus QP.

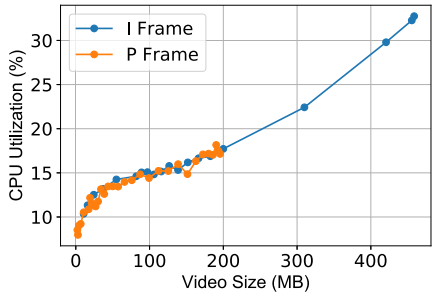


Fig. 6. CPU utilization versus video size.

TABLE II
THE AVERAGE CPU UTILIZATION VERSUS I/P FRAME CHANGES

Frame Type	*scheme-I*	*scheme-P*	Original
Average CPU Util. (%)	11.76	8.36	8.34
Total Video Size (MB)	60.83	12.06	9.32

the largest QP to keep the original quality. Under this condition, CPU utilization can be increased without influencing the original video quality by decreasing QP.

3) Quantitative Validation: Settings. To quantify the performance of changing frame type, we re-encode a 1-min video with X264 [17]. The original video is composed of 14 I frames, 572 P frames and 1214 B frames respectively. We use `-qpfile` to change frame types and `-crf` to activate CRF (Constant Rate Factor) mode to guarantee the video quality [18]. For convenience, we name the scheme by changing all frames to I/P frames *scheme-I* and *scheme-P*. We use GOM Player running on a PC (i5-4200U CPU with 2 cores and 4 threads, 8G RAM, Windows 10 17134.1) to play the re-encoded video and record the CPU utilization and video file size.

Frame Type versus CPU Utilization. The result of *scheme-I*, *scheme-P*, and the original video is shown in Table II. The video encoded using scheme-I is all composed of I frames while the video encoded using *scheme-P* is all composed of P frames except individual I frames as necessary reference frames. The original video consists of I, P, and B frames. We can find that: (1) *scheme-I* gains around 3.4% higher CPU utilization than *scheme-P*, and there is no significant difference in CPU utilization between *scheme-P* and the original video; (2) the video size increases in both cases, and *scheme-I* has a much larger increment. Therefore, we conclude that changing frames to I can both increase CPU utilization and video size while changing frames to P only increases a little video size and has almost no effect on CPU utilization. Consequently, changing frame type from P or B to I is a feasible way to increase CPU utilization while decoding but the amount of change is limited.

QP versus CPU Utilization. We quantify the relationship between QP and CPU utilization under both *scheme-I* and *scheme-P*, which is shown inFig. 4. Under CRF mode, if the QP value of frame i is not specified, it will be set as QP^i_{crf} according to the CRF mode parameter (default 23). Assuming that there are N frames in the video, the average QP of the video under CRF mode is

$$QP^{avg}_{crf} = \frac{1}{N}\sum_{i=1}^{N} QP^i_{crf}, \tag{3}$$

which is denoted by a vertical line in Fig. 4. Note that the video quality will not be affected only when QP is less than QP^i_{crf}. We vary QP from 0 to 50, and the result shows that CPU utilization increases nearly in a linear way for *scheme-P*. While for *scheme-I*, the linear relationship between QP and CPU utilization appears separately when $QP > QP^{avg}_{crf}$ and $QP \le QP^{avg}_{crf}$. Moreover, QP modification under *scheme-I* brings larger CPU utilization change when $QP \le QP^{avg}_{crf}$.

CPU Utilization versus Video Size. It is worth mentioning that the reduction of QP brings an increase in video size under both *scheme-I* and *scheme-P* frame scenarios. For example, the original 9.32 MB video can be increased by several times. Comparing Fig. 5 to Fig. 4, we find that though I frame gains more significant CPU utilization change by changing QP, it is at the cost of larger video size. Fig. 6 depicts the relationship between video size and CPU utilization. The conclusion is that CPU utilization is fundamentally determined by video size, i.e., video bit rate, and no significant difference exists between I frames and P frames.

Algorithm to Change CPU Utilization. Without loss of generality, denote the CPU utilization under *scheme-P* and *scheme-I* $U_P(qp)$ and $U_I(qp)$ when $QP = qp$. Obviously, $U_I(qp)$ should be larger than $U_P(qp)$. However, whether $U_I(QP^i_{crf}) < U_P(0)$ or not is uncertain and thus we have two cases:

$$U_P(QP^i_{crf}) < U_I(QP^i_{crf}) \le U_P(0) < U_I(0) \tag{4}$$

$$U_P(QP^i_{crf}) < U_P(0) < U_I(QP^i_{crf}) < U_I(0) \tag{5}$$

The case denoted by (4) is shown in Fig. 4, and (5) is the other case where the CPU utilization of I frame and P frame are not overlapped when $QP < QP^i_{crf}$.

Given a designed CPU utilization as U_{design}, the frame type change and QP value decision for encoding a frame can be calculated as following:

First, U_{design} is compared to $U_P(QP^i_{crf})$ and $U_I(0)$. If $U_{design} < U_P(QP^i_{crf})$, then we use *scheme-P* with QP^i_{crf} to ensure that the video quality does not decline. If $U_{design} > U_I(0)$, which means that U_{design} is beyond the maximum CPU utilization we can reach, the frame will be encoded using *scheme-I* with $QP = 0$.

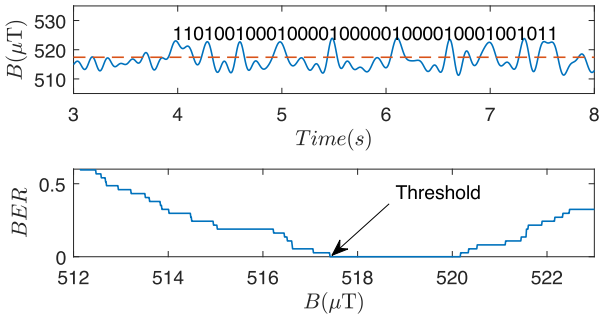


Fig. 7. Threshold tuning when receiving a preamble. B_{th} with minimum preamble decoding BER is chosen.

If the above conditions are not met, U_{design} is compared to $U_P(0)$ and $U_I(QP^i_{crf})$. If $U_{design} \geq U_I(QP^i_{crf})$, then we use *scheme-I* with $QP = U_I^{-1}(U_{design})$.[1] Otherwise, if $U_{design} \leq U_P(0)$, then we use *scheme-P* with $QP = U_P^{-1}(U_{design})$. If not, i.e., the case of (5) appears and $U_P(0) < U_{design} < U_I(QP^i_{crf})$, we prefer to use *scheme-I* with QP^i_{crf} as I frame benefits the video playing.

B. Transmission Design

In this section, the data frame design and data modulation scheme are introduced. The data frame consists of the preamble followed by the payload. The preamble field is used for synchronization and parameter tuning.

1) Preamble Design: The preamble is used to synchronize the receiver with the sender. For synchronization, a template is generated and cross-correlation is performed on the received magnetic signal on all X, Y, Z axis respectively. The axis with the highest correlation coefficient peak is used for synchronization. Besides synchronization purpose, in `MagView++` the preamble also serves as parameter tuning for the receiver to set the demodulation threshold B_{th}, which is an important parameter in the ASK modulation and introduced later.

Intuitively, the length of the preamble should guarantee stable synchronization and accurate parameter estimation. In `MagView++`, we empirically investigate and use a 37-bit-long preamble followed by a 300-bit payload. The designed preamble and parameter tuning process can be found in Fig. 7.

With a data frame consisting of the preamble, the payload and FEC, we now introduce how to modulate the data frame on CPU utilization changes.

2) ASK Modulation: Common digital modulation schemes include amplitude-shift keying (ASK), frequency-shift keying (FSK) and phase-shift keying (PSK). ASK uses different amplitudes to represent digits (or symbols), i.e., a high-amplitude signal represents "1" and a low-amplitude signal for "0". `MagView++` employs ASK modulation due to this property.

Specifically, we use 2-ASK which is the simplest ASK for robust data transmission facing the weak magnetic signal with ambient inference. In 2-ASK, we define two levels of CPU utilization for each frame: the low-level U_l and the high-level U_h. U_l equals to the CPU utilization under *scheme-P* (all other frames are transformed into P frames) with QP automatically assigned by the encoder, i.e., $U_P(QP^i_{crf})$. For U_h, we let $U_h = \alpha U_l$.

Essentially, `MagView++` makes use of the CPU utilization margin to embed information. As a result, the available margin is limited by the capacity of the CPU, the video itself (e.g., format and size) as well as background applications that use the CPU. Let U_{back} denote the sum of CPU utilization of the background applications and U_{video} stand for the CPU utilization of the video without re-encoding, then the margin of available CPU utilization we can use is:

$$U_{margin} = 100\% - U_{back} - U_{video} \tag{6}$$

Actually, the CPU model of the device that plays the malicious video is unknown and it is infeasible to estimate U_{back}. Therefore, we can only estimate U_{video} according to Fig. 4 with the knowledge of QP and assumptions about CPU types. The background CPU utilization U_{back} is assumed to be a constant value. This is reasonable in some scenarios such as surveillance camera system where the computer mainly runs a displaying task.

The encoder will choose an appropriate α to calculate U_h. Then the encoder uses the Algorithm in Section IV-A3 to calculate the frame type and QP to derive the designed CPU utilization U_{design}. In our implementation, we simply set $\alpha = 5$ to ensure sufficient discrimination between the magnetic signal emitted by CPU module under U_h and U_l. On the receiver side, utilization is decided by comparing with a threshold value of magnetic induction intensity of the CPU module, i.e., B_{th}.

3) DSSS-like Bit Encoding: With the two levels U_l and U_h, we can simply encode "1" and "0" on the two levels. However, in practice, this is error-prone because CPU utilization changes need a response time and it cannot change sharply. Therefore, to minimize error and enhance robustness, we employ a DSSS-like[2] bit encoding scheme.

We encode a single bit with several sequential changes in CPU utilization. Define the number of frames that represent a single bit as T_B, hereafter we name it to code element length. A little T_B means a higher bit rate. However, the smaller the T_B value, the harder it is to distinguish it from the magnetic signal. Therefore, we re-encoding the video by using different T_B values and found that $T_B = 3$ is the minumumn value at which "0" and "1" can be distinguished from the received magnetic field signal.

To minimize the increase in video size and cpu utilization caused by embedding sensitive information, we only change the frame type and QP of one frame within T_B frames. In other words, we use "*low-high-low*" to encode "1" and "*low-low-low*" for "0". Low CPU utilization is preferred to keep the covert channel stealthy. Fig. 8 shows an example of encoding bit with the DSSS-like bit encoding scheme. The second row of Fig. 8 is the received raw signal shown in the first row. Note that the high-frequency noises exist and a low pass filter should be used to filter out the high-frequency noise.

[1] $U_I^{-1}(\cdot)$ is the inverse function of $U_I(\cdot)$.

[2] The word DSSS (Direct Sequence Spread Spectrum) means to encode a symbol (4 bit) onto a 32-bit long sequence in 802.15.4 standard.

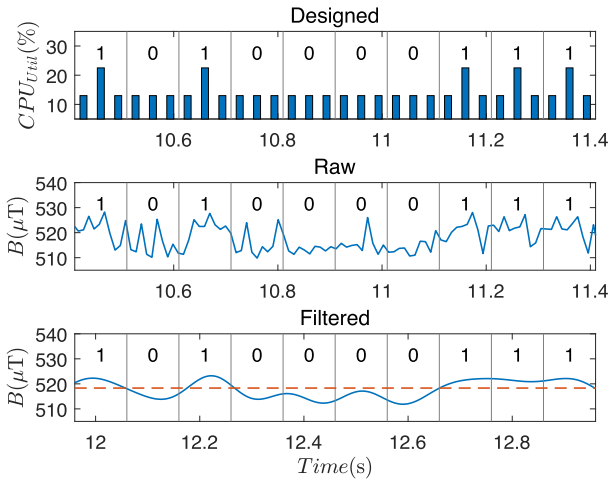


Fig. 8. DSSS-like bit encoding scheme. "*low-high-low*" means "1" and "*low-low-low*" stands for "0".

4) Throughput Analysis: The principle of MagView++ is to manipulate the computation of video frames to increase the resulting CPU utilization when decoding. As MagView++ encodes bits on the granularity of the frame, thus its maximum transmission speed is limited by the frame rate of a video, denoted as FPS.

The transmission speed of MagView++ is also decided by the number of available levels, denoted as N_{level} as well as the DSSS-like encoding scheme, and we formulate it as:

$$Speed_L = \frac{FPS \times log_2 N_{level}}{T_B}. \quad (7)$$

For example, in the current implementation, N_{level}=2, T_B=3, and FPS=30, the expected transmission speed is 10 bps. With a 37-bit long preamble and a 300-bit long payload, the expected throughput is 8.9 bps. In Section. V, we demonstrate that MagView++ can achieve this value with only 0.0057 BER even when FEC is disabled.

C. Receiver Design

At the receiver side, signal pre-processing, preamble detection and parameter tuning (magnetic induction intensity threshold B_{th} for decoding) are designed.

The received magnetic signal is first pre-processed by a low pass filter with a sliding window to filter out the high-frequency noises. In our implementation, we choose to use a finite impulse response low pass filter, with a cutoff frequency f_c and an filter order N to be determined. F_c determines the frequency range of the retained signal, while N determines the slope of the filter response curve. In current implementation, the frame rate $FPS = 30$ and $T_B = 3$. Therefore, the fundamental frequency of the transmitted signal is 10 Hz. We adjust f_c from 10 Hz and choose an appropriate value of N to minimize the bit error rate (BER) obtained by decoding on the same signal. Finally, we choose $f_c = 15.7$ Hz and $N = 3101$. The signal after filtering is depicted in the bottom row of Fig. 8.

Then cross-correlation is conducted between the filtered signal and the template along all three axes. The axis with the highest correlation coefficient is chosen as the axis of the covert channel signal.

Besides, parameter tuning is performed to derive a proper B_{th}. Specifically, we increase the threshold value and decode the preamble signal. With the increase of tested thresholds, the resulting decoding BER is first decreasing to a minimum value (e.g., 0) and then keeps stable and then increases, as is shown in Fig. 7. Empirically, we choose the threshold value with minimum BER.

V. Evaluation of MagView++ for Smartphone Receivers

In this section, we first prototyped the Frame-type-and-QP-based scheme of MagView++ for smartphone receivers to test its overall performance and then evaluated the impact of different factors.

A. Experiment Setup and Performance Summary

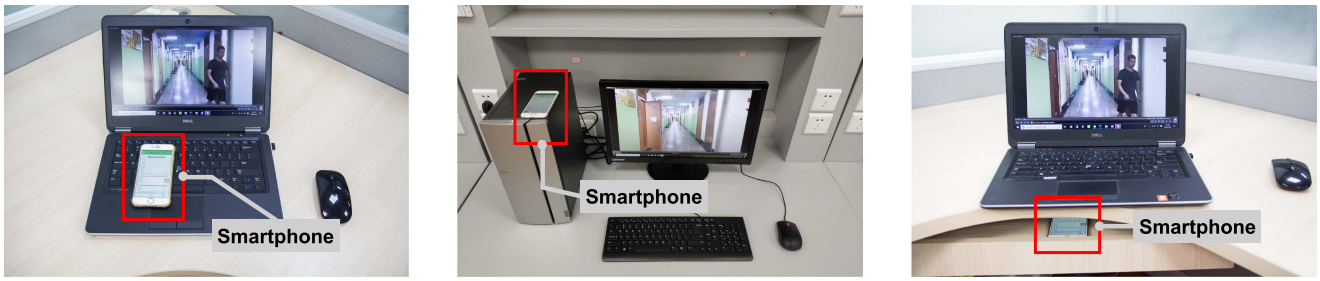
We utilize a real surveillance video [20] downloaded from Youtube. The video is with 1920x1080 resolution, 30 fps, and 1642 kbps bit rate. For simplicity, we used x264 [17] to re-encoded the video offline to embed sensitive information. It is possible to online encode the video on a surveillance device as the hardware performance continues to improve. During the re-encoding process, U_l and U_h were 16.325% and 81.8% respectively to ensure sufficient discrimination between the magnetic signal emitted by the CPU module under U_h and U_l. We embedded 1.5 Kb data into the video, and the bit rate increased to 15130 kbps consequently. We used an iPhone 6 with its built-in magnetometer to collect the magnetic signals from a Dell E7440 laptop with Intel i5-4200U Processor, as shown in Fig. 9(a).

Results. At the receiver side, after demodulation and decoding, we calculated statistically the bit error of the transmitted data. Results show that MagView++ can achieve the theoretical 8.9 bps with 0.0057 BER, which means that it takes only 15 seconds to transfer a 128-bit key.

B. Impact of Various Settings

In this subsection, we evaluated MagView++ for smartphone receivers in various settings, including *background applications, transmitters, receivers, sender-receiver distances, video players, and surroundings*. Unless otherwise stated, the experiment setup in Table III was used in all experiments. We also used a video taken by iPhone 7P in the corridor with the same 1920x1080 resolution, 30 fps as the surveillance video from Youtube [20], and 10594 kbps bit rate considering that the video is compressed by Youtube and therefore it is different from its original version. The metric we focused on was bit error rate (BER) instead of transmission speed as a covert channel and therefore the following experiments were all revealed by BER.

1) Background Application: In this experiment, Chrome, Word, and Microsoft Terminal Server Connection (MSTSC)



(a) Laptop-smartphone. (b) PC-smartphone. (c) Smartphone under a desk.

Fig. 9. Experiment setup of MagView++ for smartphone receivers under different scenarios. A video with embedded data is played on laptops or desktop PCs. The magnetic signal emanated from the CPU module is collected by a smartphone with its built-in magnetometer.

TABLE III
THE FUNDAMENTAL SETTINGS IN THE EXPERIMENTS OF MagView++ FOR SMARTPHONE RECEIVERS

Settings	Description
Video reproduction tool	FFMPEG [19]
Encoder	X264 [17]
Original video	A 1920x1080 30 fps surveillance video
Re-encoded video	Malicious: with 1.5 Kb data embedded
Video frame rate	30 fps
Operating system	Windows 10 17134.1
Sensor sampling rate	100 Hz
Video player	GOM Player
Code element length	3 frames
Transmitter	DELL e7440
Receiver	iPhone 6 with its built-in magnetometer
Metric	BER without using FEC

TABLE IV
AVERAGE CPU UTILIZATION AND BER VERSUS DIFFERENT BACKGROUND APPLICATIONS

APP	U_{total} (%)	U_{player} (%)	U_{back} (%)	BER
None	29.3	23.7	5.6	0.0057
Chrome	31.7	22.9	8.8	0.0157
Word	30.6	23.1	7.5	0.0095
MSTSC	29.8	23.1	6.7	0.0072
ALL	34.3	23.3	11.0	0.0238

were used as background applications considering they are common working applications in an office computer. In the "None" case, the video player was the only running application. In the "Chrome" case, ten tabs of different news sites were opened. In the "Word" case, five Word windows were opened and each contains at least one page of content. In the "MSTSC" case, the computer as the transmitter was connected to another computer by MSTSC. And in the "ALL" case, all of the above background applications are running simultaneously to mimic a real application scenario. We used *psutil* [21] to record the total CPU utilization U_{total} and the CPU utilization of the video player U_{player}, then the CPU utilization of the background application was $U_{back} = U_{total} - U_{player}$. Table IV shows the results. The BER increases a little when there is some background application. However, compared to the situation where there is no background application, the performances are still good with BER all lower than 0.05, which is in line with the expectation of the experiment. It has to be said that high CPU utilization processes such as other video playing, decompression operations, etc. can cause MagView to fail, but this is not often the case when the video is playing in the foreground.

2) Transmitter: We used 9 different computers as transmitters to test their influence on BER, which were DELL e7440 (i5-4200U), DELL xps14 (i7-3537U), DELL xps13 (i5-6300U), Lenovo g40 (i5-5200U), Lenovo Zhaoyang g42-80 (i3-7100U), Lenovo r720 (i5-7300H), Dell inspiring 14 (i5-8250U), PC1 (i5-8400), PC2 (i5-3470T) respectively. The experimental setup of PC1 is shown in Fig. 9(c). We respectively found a relatively better location to set the receiver so that the receiver could record strong signals for each computer. The results are shown inFig. 10. Except for the two desktop computers PC1 and PC2, the BERs for all other computers are lower than 0.1. One of the explanations of the desktop case is that the distance from the receiver to the CPU is larger than that of the laptop cases. Nevertheless, the BER of the two desktop PCs can be reduced to below 0.03 by using Hamming FEC with alphabet size $r = 2$, at the cost of a bit rate decrease to 3.0 bps.

3) Receiver: We used 6 smartphones, 1 smartwatch and a data acquisition (DAQ) device [22] connected with a low-cost DRV425 [23] magnetic sensor as the receivers to record the magnetic signals. The DAQ device gains high sampling rate (200 kHz) than others. The results are shown in Fig. 11. Except for iPhone 7P and Vivo, smartphones work well with the BER lower than 0.1. The reason why Vivo has a poor performance, as we infer, is that the sampling points are uneven. The DAQ receiver demonstrates the lowest BER due to its high sampling rate. In addition, Huawei Watch 2 has a good performance with a BER of 0.02, which shows the feasibility of using a smartwatch to launch an attack.

4) Sender-receiver Distance: As we mentioned above, distances between the receiver and the CPU can make a difference to the results, so we put the receiver (the iPhone 6) at different distances from the transmitter where the malicious video is played. The results in Fig. 13 show low BER when the distance is below a value, say 6 cm with BER lower than 0.1. Moreover, we put the iPhone 6 under the transmitter computer separated by a wooden shelf as shown in Fig. 9(c), in which situation the distance between the iPhone 6 and the bottom of the transmitter is about 4 cm with 0.065 BER. This illustrates that wooden shielding has little influence on magnetic signals. Even though the distance is relatively short in the current implementation,

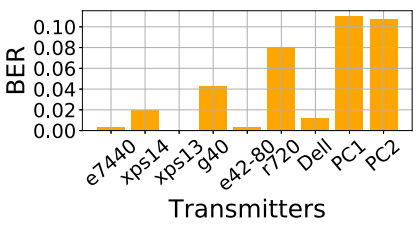


Fig. 10. BER versus different transmitters.

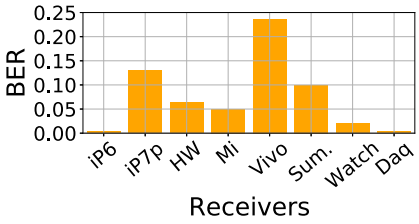


Fig. 11. BER versus different receivers.

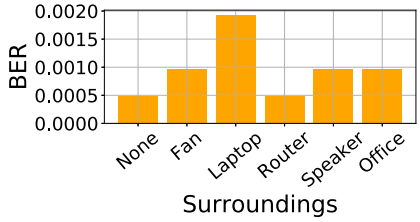


Fig. 12. BER versus different surroundings.

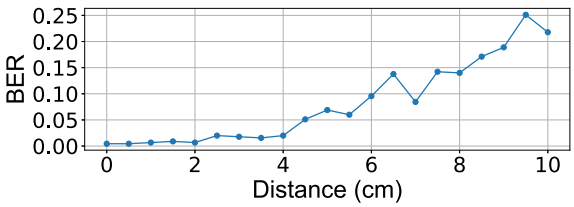


Fig. 13. BER versus different distances.

TABLE V
BER VERSUS DIFFERENT VIDEO PLAYERS

Video Players	Default Decoder	BER (SW)	BER (HW)
Movies & TV	HW	N/A	Failed
VLC Player	HW	0.0100	Failed
KMPlayer	SW	0.0167	Failed
GOM Player	SW	0.0033	Failed
MPC-HC	HW	0.0033	Failed
Pot Player	SW	0.0067	Failed
UMPlayer	SW	0.0033	N/A
SMPlayer	SW	0.0200	Failed
Aiqiyi	SW	0.0233	Failed
QQ Player	HW	0.0033	Failed
Xunlei Player	SW	0.0067	Failed
Baofeng	SW	0.0033	Failed

we believe it can be extended by a dedicated device with more sensitive sensor and provide a theoretical analysis in the Discussion Section.

5) Video Player: We selected 12 common video players as shown in Table V and installed them on the DELL E7440 Laptop. Of the 12 video players, 8 video players use software-based decoding method (SW) by default while the remaining 4 video players use hardware-based decoding method (HW) by default. We tested all the 12 video players with the malicious video encoded by the ASK modulation and the results are shown in Table V. "N/A" means that the video player does not has the corresponding decoding mode. Results show that when using software decoding, the BER of all video players is lower than 0.025. It can be inferred that the different performance between video players is due to additional computing tasks such as playlist collection or video playback optimization. However, when using hardware decoding, none of the players support the information transfer using the Frame-type-and-QP-based scheme (marked as "failed"). This is because the power consumption of hardware decoding is so low that the change of CPU magnetic field cannot be resolved. Therefore the embedded information cannot be extracted.

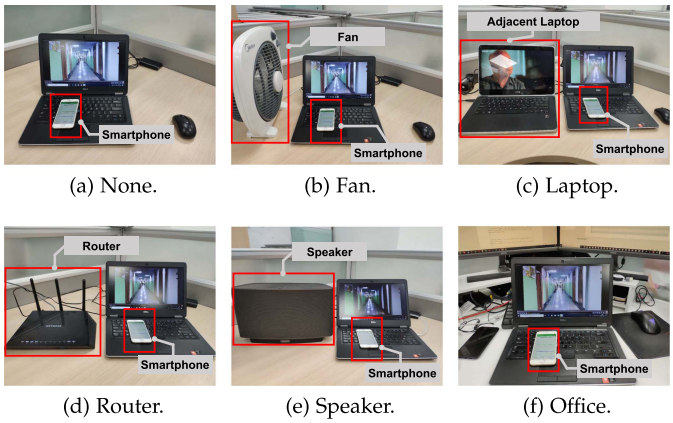


Fig. 14. Experiment setup for different surroundings. An electric appliance is placed next to the receiver, i.e., the smartphone, to test the impact of surrounding devices.

6) Surroundings: To investigate the performance in real application scenarios, we tested `MagView++` in six different surroundings as shown in Fig. 14, including (a) no adjacent device, (b) a fan nearby, (c) a laptop nearby playing a video, (d) a router nearby, (e) a speaker nearby and (f) a real office scenario with a desktop computer under a desk and an air conditioner above. The BERs in all scenarios are no more than 0.003 as shown in Fig. 12, which means there is no significant effect of adjacent devices on the BER of `MagView++`. The reason is that the strength of the low-frequency magnetic signals is inversely proportional to the distance $(1/r^3)$ from the device [24], which leads to little impact on the surrounding devices.

VI. DESIGN OF `MagView++` FOR DEDICATED RECEIVERS

In this section, we present the design of `MagView++` for dedicate receivers with high sampling rates. Different from the sampling rate on smartphones, e.g., 100 Hz, a device with 10 kHz sampling rate can record the CPU magnetic waveform of each video frame decoding. Therefore, more detailed features of the CPU magnetic field can be captured. The overall design of `MagView++` for dedicated receivers is shown in Fig. 3. For an original video, `MagView++` first changes the timestamp of each frame to achieve two different timestamp offset levels, i.e., "$nT + D$" and "$nT - D$", and then uses the two levels to embed and modulate the sensitive information into video frames.

A. Frame Display Time Control

1) Variable Frame Rate Supports Frame Timestamp Changing: Constant frame rate (CFR) video uses a timer to

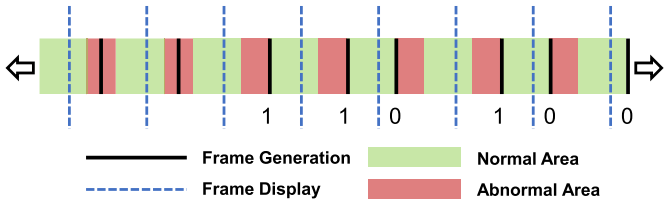


Fig. 15. Video frame rate versus screen refresh rate.

display each video frame, and the period of the timer is fixed. For example, a 30 fps video plays 30 frames evenly in one second. On the contrary, variable frame rate (VFR) allows for the frame rate to change actively during video playback [25]. VFR is especially useful for creating videos of slideshow presentations, videos with large amounts of completely static frames, or if the video contains a combination of 24/25/30/50/60 FPS footages and the creator or editor of the video wishes to avoid artifacts arising from framerate-conversion [25]. To achieve VFR, each video frame has a timestamp, and the video player uses the timestamp to determine the moment at which each frame is displayed. In Section II-A we mentioned that the moment of decoding another video frame is determined by the moment of the video frame playing in the queue. This allows embedding sensitive information into the timestamps of video frames.

2) Using Different Timestamp Offset to Embed Data: Supposing that the original video has a frame rate of F_{PS}, then the distance between two frames is $T = \frac{1}{F_{PS}}$. We create two offset levels for data embedding: $nT + D$ and $nT - D$, where nT is the display time of frame n, D is the offset, and $D < \frac{2}{T}$. Then, we use the two offset levels to embed information as Pulse Position Modulation (PPM). Specifically, we use a frame with display time $t = nT + D$ to represent a "1" bit, and use a frame with display time $t = nT - D$ to present a "0" bit.

3) Invisibility of the Timestamp-based Scheme: To ensure the invisibility of the information transmission, we investigated whether adding an offset to the timestamp of the frame would be reflected in the viewing effect of the video. We first did a theoretical analysis and found that changing the timestamp might have a small probability of causing frame loss. Then, we experimented and proved that the conclusion of the theoretical analysis was correct. At last, we did a user study and found that most of the volunteers felt that the video with information embedded was not different from the original video.

Theoretical Analysis. Fig. 15 demonstrates the cause of video frame loss. The solid black line represents the moment when the video is decoded, which is determined by the timestamp. The blue dashed line represents the moment when the monitor displays a frame. By changing the moment when each frame is displayed, the frames of the video that are displayed evenly will become uneven, with some frames being brought forward and others delayed. However, the monitor's refresh rate is constant, and typically the refresh rate is 60 Hz. When the video player needs to display a frame, it first sent the frame to be displayed to the buffer of the monitor. When the monitor refreshes, it takes the frame from the buffer and displays the frame. Therefore, if there are two frames generated between two frames displayed on the monitor, i.e., two solid black lines appear in the middle of the two blue dashed lines, only the later generated frame will be displayed and the earlier generated frame will be discarded. Note that the differences between the moment when video playback starts and the moment when the monitor refreshes are random. When the uniformly distributed blue dashed line falls into the green area, the timestamp offset of frames does not affect the video playback. On the contrary, when the blue dashed line falls into the red area, the frame represented as a "1" before the frame represented as a "0" will be discarded.

We use T to represent the time between the video start playing and the display refresh, and $T \in (0, \frac{1}{R})$, where R is the refresh rate of the monitor. We suppose that the distribution of T is uniformly distributed, i.e., it is equally likely to start playing at any time. Denote the probability of frame discarding as P, then we have

$$P = P(10) \cdot 2D \cdot R = \frac{1}{2} DR, \tag{8}$$

where $P(10)$ is the probability of bit "10" occurrences and $P(10) = 0.25$. For example, when $R = 60\,\text{fps}$, $D = 3\,\text{ms}$, the probability of frame discarding is 9%, which is acceptable as insufficient CPU computing capacity can also cause frame loss.

Note that the above analysis are based-on the assumption that the video player does not use frame-skipping or delaying techniques to improve the playback performance. Both frame-skipping and delaying will make the Timestamp-based scheme fail.

Verified by Video Recording. We conduct an experiment to verify this model. We generate 10 pictures with numbers 0-9 respectively. Then, we take each image as a frame and combine them into a video. Therefore, the numbers 0 to 9 loop through the video. We set the frame rate of the video as 60 fps. Then, we embed random information in the video with $D = 3\,\text{ms}$ respectively. We play the video 10 times using a monitor with 60 Hz refresh rate and using an iPhone 12's 240 fps camera to decode what the monitor shows. The numbers of the lost frame and the total frame of each trace are 7/32, 0/33, 6/33, 0/33, 4/32, 8/31, 0/33, 10/34, 0/34, 0/33, respectively, and the average frame losing rate is 10.7034%. This result is close to the 9% frame loss rate that we theoretically calculated, verifying the correctness of the above theoretical analysis. Note that video frame loss may occur on a normal video when CPU computing resources are occupied by other applications or the video decoding is too computationally intensive. Therefore, we believe most people will not suspect the video is malicious, and we did a user study to prove it.

Verified by User Study. To verify that the frame loss has no significant effect on the view, we conduct a user study with 44 volunteers. We follow the local regulations to protect the rights of human participants despite the absence of the Institutional Review Board (IRB). We select 3 different videos with different FPS, i.e., 24 fps, 30 fps, and 60 fps respectively. For each video, we let the volunteer view an original video and video embedded data using offset $D = 3\,\text{ms}$. Taking video *The Hobbit* as an example, 4 out of 44 persons said that the original video and the video with data embedded were different. However, only 2

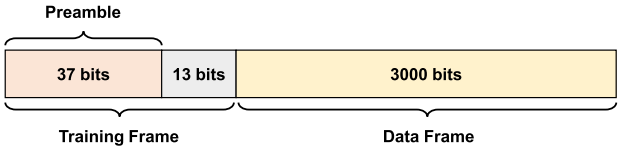


Fig. 16. Data Frame of Timestamp-based Scheme.

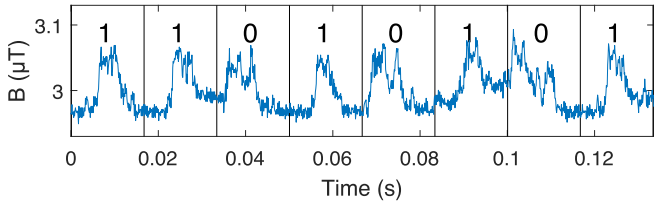


Fig. 17. Magnetic field waveform of the PPM. The rising edge of a frame coded "1" appears later than that of a frame coded "0".

out of the 4 persons correctly identified the original video. The 4 persons all said that the difference between the 2 videos was lag. Note that even though the video is somewhat lagging, the victim may think the lag is coming from other reasons such as insufficient CPU performance. Therefore, we conclude that the Timestamp-based Scheme is implicit.

B. Transmission Design

Fig. 16 shows the data frame design of Timestamp-based Scheme. The data frame contains a 37-bit preamble frame, a 50-bit training frame, and a 2950-bit data frame. Among them, the first 37 bits of the training frame are shared with the preamble frame. The preamble is used to synchronize the receiver with the sender, while the training frame is used to train an arbiter for further decoding the data frame.

1) Preamble and Training Frame Design: Similar to the Frame-type-and-QP-based scheme, the preamble is used to synchronize the receiver with the sender. We use the same 37 bits as that used in Frame-type-and-QP-based Scheme. A template is generated and cross-correlation is performed on the received magnetic signal.

For the receiver to be able to distinguish between bit "0" and "1", we design a 50-bit long training frame, where the first 37 bits are reused in the preamble. The length of the training frame is a trade-off between the robustness of the receiver training and the transmission efficiency.

2) Pulse Position Modulation: By changing the timestamp of each frame, the time at which the CPU decodes that frame can be changed. The magnetic waveform for decoding each frame can be seen as a pulse, and thus the information can be encoded using Pulse Position Modulation (PPM). As is stated in Section VI-A2, we have two timestamp offsets, i.e., $+D$ and $-D$. We use $+D$ to represent a "1" and use $-D$ to represent a "0". Therefore, each frame of a video can carry a bit of information. Fig. 17 shows the magnetic induction intensity near the CPU when PPM is adopted. We can see that frames encoded as "1" are decoded later, while frames encoded as 0 are decoded earlier.

Supposing that the bit rate of Timestamp-based Scheme is $Speed_H$, then we have:

$$Speed_H = \frac{2950}{3000} FPS. \tag{9}$$

For example, if the frame rate $FPS = 60$, the bit rate is 59 bps. By using dedicated receivers with higher sampling rates, the bit rate of Timestamp-based Scheme is more than 6 times the bit rate of Frame-type-and-QP-based Scheme.

C. Receiver Design

We used a low-cost data acquisition (DAQ) device (less than $100) [26] connected with a DRV425 [23] magnetic sensor (less than $30) as the receiver to record the magnetic signals. The sampling rate of the DAQ device is set to 10 kHz, which is enough for the Timestamp-based scheme. The receiver first uses a sliding window combined with cross-correlation with the template preamble sequence, and regards the point corresponding to the maximum cross-correlation value as the starting point of the transmission sequence.

Then, the receiver uses the training frame to train an arbiter for further decoding the transmitted "0"s and "1"s. Supposing that the start time of the first bit "1" is T_{start}, we segment the magnetic sequence as $\left[T_{start} - 2D + \frac{k-1}{FPS}, T_{start} - 2D + \frac{k}{FPS}\right)$, where $k = 1, 2, ..., 3000$. The first 50 traces are used for training a classifier, and then we use the classifier to classify the remaining 2950 traces. Because the difference between bit "0" and bit "1" lies at the moment when high CPU utilization occurs, that is, the energy distribution is different, we use "energy ratio by chunks" as the feature to classify the traces. Specifically, we first normalize the sequence into $[0, 1]$. Then, we evenly divide the trace into 10 pieces and calculate the ratio of the sum of squares of each piece to the total sum of squares. This gives each trace 10 features that can be used for classification. We use Support Vector Machine (SVM) as a two-class classifier to classify each received bit.

VII. Evaluation of MagView++ for Dedicated Receivers

In this section, we first prototyped MagView++ for dedicated receivers with the Timestamp-based scheme to test its overall performance, and then evaluated the impact of various factors.

A. Experiment Setup and Performance Summary

We utilize a real surveillance video [27] download from Youtube. The video is with 1280x720 resolution, 60 fps, and 5942 kbps bit rate. For simplicity, we used *mkvtoolnix* to modify the timestamp of each frame to embed sensitive information. It is possible to online encode the video on surveillance devices as modifying the timestamp does not require the computation of video encoding and decoding. We used offset $D = 3\,\text{ms}$ and embedded 2.95 Kb data into the 1 min video for five times. During this process, the bit rate of the video does not change. We use DRV425 magnetic sensor [23] and Art Technology USB3200 data acquisition device [26] to collect the magnetic

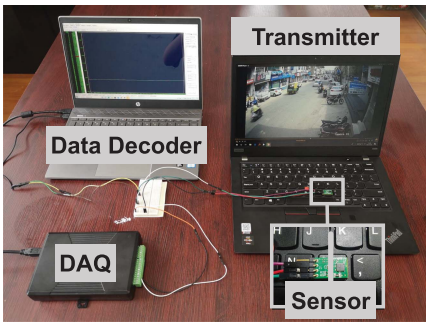


Fig. 18. Experimental setup of MagView++ for dedicated receivers. A video with embedded data is played on laptops or desktop PCs. The magnetic signal emanated from the CPU module is collected by a data acquisition device (DAQ) with a magnetic sensor.

TABLE VI
THE FUNDAMENTAL SETTINGS IN THE EXPERIMENTS OF MagView++ FOR DEDICATED RECEIVERS

Settings	Description
Video reproduction tool	FFMPEG [19]
Encoder	mkvtoolnix [28]
Original video	A 1280x720 60 fps surveillance video [27]
Re-encoded video	Malicious: with 2.95 Kb data embedded
Video frame rate	60 fps
Operating system	Windows 10 17134.1
Sensor sampling rate	10 kHz
Video player	GOM Player
Frame Timestamp Offset	$\pm$ 3 ms
Transmitter	Thinkpad X395
Receiver	magnetometer with DAQ
Metric	BER without using FEC

signals from a Thinkpad X395 laptop with AMD Ryzen 7 3700U Processor, as shown in Fig. 18.

Results. At the receiver side, after demodulation and decoding, we calculated statistically the bit error rate (BER) of the transmitted data. Results show that the Timestamp-based scheme can achieve the theoretical 59 bps with 0.0025 average BER.

B. Impact of Various Settings

In this subsection, we evaluated MagView++ for dedicated devices with Timestamp-based scheme in various settings, including *video contents, frame rates, transmitters, video players, and sender-receiver distances.* Unless otherwise stated, the experiment setup in Table VI was used in all experiments. The metric we used for Timestamp-based Scheme is also the BER.

1) Timestamp Offset: We used 1 ms to 5 ms offset to encode data into the video according to Table VI. The selection of offset D is a trade-off. A small offset means that the frame losing rate of encoded video can be smaller according to Equ. 8, resulting in higher invisibility. However, A small offset will result in a larger BER as it is difficult to distinguish "0" and "1". On the contrary, A larger offset means a higher frame losing rate and lower BER.Fig. 19 shows the results. The receiver failed to decode the transmitted data when $D = 1\,\text{ms}$, and the BER is high at $D = 2\,\text{ms}$. When $D \geq 3\,\text{ms}$, the BER is acceptable. Therefore, we choose $D = 3\,\text{ms}$ in Timestamp-based Scheme to encode data into videos.

2) Frame Rate: We re-rendered the video in Table VI to 24 fps, 30 fps, and 50 fps respectively to encode data. The timestamp offset D of the videos are 7.5 ms, 6 ms, and 3.6 ms respectively, ensuring that the offset is constant in proportion to the length of a frame. Since the length of the video remains the same (1 min), the bits of data to be transmitted are 1200, 1500, and 2500 bits respectively. We compare the bit error rates of the three re-rendered videos with the original 60-frame video, and the results are shown in Fig. 22, where the BERs are all below 0.0025. In general, the higher the frame rate, the higher the BER tends to be. Since MagView++ can be interfered by other processes, the BER has a certain randomness. We also test the BER for a 120 Hz video but the magnetic signal cannot be decoded, this is because some video frames cannot be decoded on time due to the insufficient computing power of video decoding.

3) Video Contents: We select 5 videos to evaluate Timestamp-based Scheme on different video contents. The No.1 video is the surveillance video in Table VI. The video numbered 2-5 were downloaded from Youtube, which are No.2: a video taken by a smartphone, No.3: a Movie, No.4: Game live, and No.5: Sports highlights. All the 5 videos are 60 fps, and the resolution are all adjusted to 1280x720 to be consistent. We took the first 1 min of all the videos for data embedding. Fig. 21 shows that the BERs of all 5 video types are lower than 0.015, this means that the Timestamp-based scheme supports various types of video contents. The result also shows that the BERs of the Game live video and the Sports highlights video are higher than others. To explain this phenomenon, we further analyze the location in the video where the error code occurs and find that when there is a large fluctuation in the bit rate of the video, such as when a still frame suddenly becomes a moving frame, its magnetic field waveform also changes, making the trained classifier unable to classify correctly resulting in error bits.

4) Transmitter: We used 5 different computers as transmitters to test their influences to BER, which were Dell Latitude E7440 (i5-4200U), Thinkpad X395 (Ryzen7-3700U), Lenovo Xiaoxin Air 15 (Ryzen7-4800U), Thinkpad X1 Carbon (i7-1135G7), and a PC (i5-3470T). For each computer, we placed the magnetic sensor where the magnetic field signal of the CPU was strongest and did not open the case of the PC. The results are shown in Fig. 23. Except for the PC, the BERs for all other computers are lower than 0.05. We believe the performance of the PC can be improved by using a DAQ with a higher resolution DAC inside.

5) Video Player: We use 12 different video players that are same as those in Section V-B5 to play the malicious video encoded with the Timestamp-based scheme. As shown in Table VII, we test the BER of the Timestamp-based scheme on all the 12 video players with hardware decoding (marked with HW) and software decoding (marked with SW) respectively. The experimental setup except for the video player is according to Table VI. "N/A" means that the video player does not has the corresponding decoding mode. "*" means that due to the slow decoding speed of the video player, there will be a delay in decoding a video with 60 fps, so a video with 30 fps is used for evaluation. The results show that when using software decoding, the BERs of most video players are less than 0.05. However,

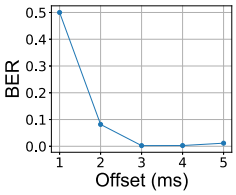


Fig. 19. BER versus different offsets.

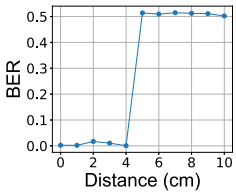


Fig. 20. BER versus different distances.

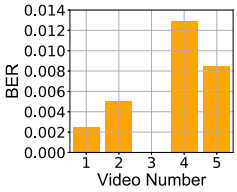


Fig. 21. BER versus different video content.

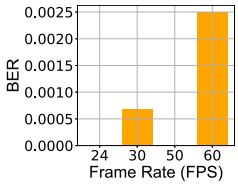


Fig. 22. BER versus different frame rates.

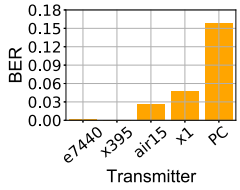
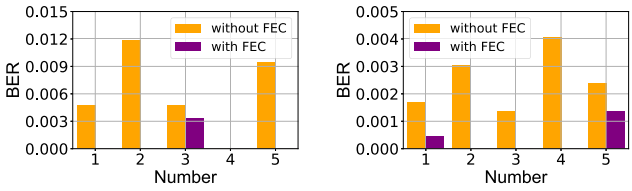


Fig. 23. BER versus different transmitters.

TABLE VII
BER VERSUS DIFFERENT VIDEO PLAYERS

Video Players	Default Decoder	BER (SW)	BER (HW)
Movies & TV	HW	N/A	Failed
VLC Player	HW	Failed	Failed
KMPlayer	SW	0.0159	0.1410
GOM Player	SW	0.0025	0.0200
MPC-HC	HW	0.0105	0.0814
Pot Player	SW	0.0062*	0.0338*
UMPlayer	SW	0.0159*	N/A
SMPlayer	SW	0.0165*	0.1600
Aiqiyi	SW	0.0469*	Failed
QQ Player	HW	Failed	Failed
Xunlei Player	SW	0.0038*	0.1410
Baofeng	SW	Failed	Failed

∗ Evaluated using a video with 30 fps



(a) Frame-type-and-QP-based Scheme. (b) Timestamp-based Scheme.

Fig. 24. Bit error rate without or with FEC.

the transmission fails when using VLC Player, QQ Player, and Baofeng. This is because these video players do not decode frames according to the timestamps. When using hardware decoding, the BER increases significantly or even the transmission fails. The reason is that when hardware decoding is used, most of the decoding tasks are calculated by GPU, and only a few tasks are calculated by CPU that contributes to the CPU magnetic field. As a result, the transmission can still be performed but with an elevated BER due to the reduced signal-to-noise ratio (SNR). In addition, the BER varies from player to player due to the existence of different additional video screen optimization calculations running on different video players.

6) Sender-Receiver Distance: As we mentioned above, distances between the receiver and the CPU can make a difference to the results, so we put the receiver (the DAQ with DRV425 magnetic sensor) at different distances from the transmitter where the malicious video is played. The results in Fig. 20 show that when the distance is below a value, say 4 cm with BER lower than 0.1. Different from Frame-type-and-QP-based Scheme, in Timestamp-based Scheme the BER rapidly increases to 50% after 4 cm. This is because the precision of the DAQ we used is only 12 bit while the precision of the iPhone 6 is 16 bit, and when the distance is greater than 4 cm, the high level and low level of the magnetic signal cannot be distinguished. Therefore, the magnetic signal cannot be synchronized. We believe the distance can be extended by using a more accurate DAQ and provide a theoretical analysis in the Discussion Section.

VIII. Transmission Error Elimination

Transmission errors can be eliminated by re-transmission, which is a common solution in most communication systems. Re-transmission is infeasible because the magnetic covert channel is one way, i.e., communication only from the high-security party to a lower one (attacker).

In `MagView++`, we exploit a forward error correction (FEC) [29] to control the errors in data transmission at the cost of redundancy brought by the error-correcting code (ECC). The ECC tolerates a limited number of errors and corrects them instead of re-transmission. We implement ECC with Hamming code [30], which is a linear error-correcting code and can correct one-bit error in each block, without detection of uncorrected errors.

Hamming code takes $2^r - 1$ bits as a block, where $2^r - 1 - r$ bits are information and r bits are error correcting codes. We implement Hamming FEC with $r = 4$ and $n = 15$ on both two schemes to achieve a trade-off between the BER and the transmission rate. We use the experiment setup in Tables III and VI respectively. We delete some information bits to add FEC bits while keeping the total number of bits to be constant. Then we re-encode the video, play the video, and receive the transmitted bits. For each `MagView++` scheme, we repeat the experiments five times. The results are shown in Figs. 24(a) and (b). With FEC, the bit rates of the two schemes decreased from 8.9 bps and 59 bps to 6.53 bps and 43.27 bps, respectively, while the bit error rates of the two schemes decreased from 0.0057 and 0.0025 to 0.00066 and 0.000369, respectively. Although the ideal bit error rate is not achieved because the errors are not uniform, with FEC the BER is almost less than 1/6 of the BER without FEC.

IX. Discussion

A. Countermeasures

`MagView++` can be defended in several ways. The simplest way is to re-encoding the videos, but this will result in extra computational overhead. Besides, an organization can do the following.

1) Shielding and Physical Isolation: Security-aware organizations may shield the high secure computers from emitting electromagnetic signals. For instance, a Faraday cage can prevent the leakage of electromagnetic signals emanating from various computer parts including the CPU, memory, and other parts. However, the signal of the magnetic covert channel is low-frequency magnetic signal and can penetrate the Faraday cage [8], [9]. Consequently, the security-aware organizations should shield the computers with thicker metal surfaces [31] as well as extend the distance. Also, they can physically isolate the computers to eliminate physical access from attackers to receive the magnetic signals.

2) Anomaly Detection: An anomaly detection system can be used to detect the abnormal operation of computers. Common anomaly detection systems use both software-based [32] and side-channel-based [33], [34], [35] detection to monitor the CPU workload or network traffic. In principle, `MagView++` can be stealthy as it hides information in a natural task of video decoding. Moreover, as the video traffic will increase when moving objects are in the video, it is difficult to detect `MagView++` by network traffic. More powerful agents may exploit specially-designed, a machine-learning-based classification that models the video decoding process. For example, a neural network suitable for time series classification can be used, such as LSTM, to implement a binary classifier that detects the presence of malicious transmission by monitoring the real-time CPU utilization and video traffic. Under this circumstance, minor CPU changes and network traffic may be detected.

B. Limitations

As low SNR and low data rates are normally the characteristics of covert channels [36], the 8.9 bps data rate of `MagView++` for smartphone receivers and the 59 bps data rate of `MagView++` for dedicated receivers are acceptable. Apart from this, there are several limitations of `MagView++`. First, the transmitter-receiver distance is limited. Under the settings in Section V and Section VII, we achieve 0.1 BER at 6 cm using iPhone 6 and 0.006 BER at 4 cm using a DAQ device respectively. The distance is actually short for practical attacks unless the attackers can get very close to the attacked computers. Nevertheless, we envision that a larger distance can be achieved by devices with more powerful magnetic sensors. Besides, we can increase the CPU change to enlarge the transmitted signal strength. Second, Frame-type-and-QP-based Scheme increases video size which occupies more storage and network bandwidth. However, this issue has been solved in Timestamp-based Scheme. Third, when using hardware decoding, `MagView++` will fail on all video players with the Frame-type-and-QP-based Scheme and a few video players with the Timestamp-based Scheme. However, in order to ensure compatibility, third-party players usually use software decoding by default. In addition, the player comes with the system, such as the Movie & TV player (Windows 10) does not support decoding H.265 (HEVC) videos, which are widely used in surveillance cameras and are supported by `MagView++`. Therefore, `MagView++` is still available in most cases.

C. Analysis for Distance Improvement

To understand the feasibility of enhancing the transmission distance of `MagView++`, we quantitatively analyze the influencing factors including CPU change rate, CPU thermal design power (TDP), Sensor sampling rate, and sensor sensitivity.

CPU change rate directly determines the strength of the generated magnetic field. According to the conclusion in Section II-B, the distance r is proportional to third power of the current i, i.e., $r \propto \sqrt{3}I$, while the current is positive correlated to the CPU change rate $U_H - U_L$. For the sake of simplicity, we approximately have $r \propto \sqrt{3}U_H - U_L$. However, since the maximum CPU change rate is 100%, the distance cannot be enlarge indefinitely.

CPU thermal design power (TDP) is another factor that determines the strength of the generated magnetic field, which can also be calculated as $r \propto \sqrt[3]{P_{TDP}}$, where P_{TDP} is the TDP of the CPU, indicating the long full power consumption limitation.

Sensor sampling rate: when the sensor sampling rate is sufficient, i.e., 100 Hz for the Frame-type-and-QP-based scheme and 10 kHz for the Timestamp-based scheme, increasing the sampling rate will not increase the attack range. This is because the dominated limiting condition in this case is the sensor sensitivity.

Sensor sensitivity: supposing that the signal can be correctly decoded only if the high level signal and the low level signal are certain counts away from each other, we have $r \propto \sqrt[3]{2^(k_s)}$, where k is the sensor sensitivity. For example, the sensor used in the evaluation of the Timestamp-based scheme is a 12-bit sensitivity sensor, i.e., $k = 12$, and the attack range is 4 cm. So it can be inferred that when $k = 24$, the attack range can be extended to 64 cm.

X. Related Work

A. Covert Channels

Covert Channel is defined as the channel that is not intended for information transfer at all but leaks sensitive data [5]. According to the physical forms of signals, covert channels can be divided into acoustic covert channels, optical covert channels, voltage/current covert channels, electromagnetic covert channels, and thermal covert channels.

Acoustic Covert Channels. Electronic devices may produce acoustic noises while operating. Guri et al. [37] propose to control the movement of the drive arm of a mechanical hard disk to produce sound at specific audio frequencies, and thus embed data into the sound. Likewise, they use the noise of computer fans to exfiltrate data by controlling the fan speed [38]. In addition, they exploit the phenomenon that capacitors and inductors in computer power supplies can produce sound, and control the sound by controlling the CPU load to achieve data exfiltration [39].

Optical Covert Channels. The Light Emitting Diodes (LEDs) on the device can be used to exfiltrate data from air-gapped networks. The data can be encoded by brightness and hue controlling [40] as well as on-off controlling [41].

Alterations to the video frame are not perceptible to human eye due to visual holdover effect can also hide information [42]. However, the information needs to be received by camera shots, which is not allowed in some high-security organizations.

Voltage/Current Covert Channels. The voltage and current on the power line vary with the power consumption of the device, so by running different programs on the device it is possible to embed the information. Existing voltage covert channels including cross-FPGA (Field Programmable Gate Array) covert channels [43], cross-room computer-power outlet covert channels [44], and covert channels between CPU cores [45]. For current covert channels, Spolaor et al [46] propose to use the USB cable current to build a covert channel from the phone to the charger.

Electromagnetic Covert Channels. This type of covert channel can be divided into two categories according to the wavelength of electromagnetic waves: near-field (within one wavelength) and far-field (greater than one wavelength) [47]. For near-field electromagnetic covert channels, Guri et al. [8], [9] and Pan et al. [48] propose to change the CPU magnetic field by controlling the CPU load to exfiltrate data. The magnetic field source can also be power supplies [49] and mechanical hard drives [16]. Matyunin et al. [50] proposed an inner device covert channel that an attacker changes video frame type and resolution to control the magnetic field generated by the CPU, and therefore by using built-in magnetometer to achieve an App-to-App covert channel. For far-field electromagnetic covert channels, the electromagnetic field source can be USB interfaces [51] and computer memory bus [52], etc.

Thermal Covert Channels. Guri et al. [53] propose a cross-computer covert channel based on the heat generated by a computer and the temperature sensors on another computer. Although the transmission rate of the thermal covert channel channel is slow, i.e., up to 8 bit in 1 hour, and the transmission distance is only up to 40 cm, it is still possible to exfiltrate short sensitive information from the air-gapped network during a long period.

Summary. Although existing covert channels can achieve good distance and transmission rate, they all require direct control of signal emitting, e.g., controlling the CPU load to control the magnetic signals. Therefore, the malware for signal controlling may be detected by security software. In addition, the device that implants the malware and the device that leaks the data must be the same devices, which limits the feasibility of the attack. `MagView++` addresses the two issues by hiding the signal controlling process into video playing and exploiting the ability of the video to propagate though the intranet to achieving a distributed covert channel. Compared with existing covert channels, `MagView++` has better stealth and implementability.

B. Other Attacks Exploiting Physical Signals

Key Extraction Attacks. The power consumption of electronic devices can be used to extract the encryption keys. Existing power analysis methods include simple power analysis (SPA), differential power analysis (DPA), template attack (TA), and correlation power analysis (CPA) [54]. Recently, the physical signals unintentionally generated by electronic devices, such as acoustic signals, electromagnetic signals, etc., are used to extract the encryption keys in a non-intrusive and long-distance manner. Genkin et al. [55] propose a simple power analysis method based on the acoustic signals emitted by capacitors and inductors in the circuit of the encryption devices and recover a full 4096-bit RSA key within 10 meters. Camurati et al. [56] propose an key extraction attack based on electromagnetic signals and achieve full key extraction of an AES-128 implementation at a distance of 10 m.

Information Inference Attacks. The physical signals unintended generated by electronic devices can also be used for an information inference attack. Genkin et al. [57] propose an screen content inference attack based on the acoustic signals generated by the screen circuits. The content displayed on the screen is usually displayed in progressive scan, and the difference in display content will result in different ultrasonic waves generated by the screen circuits. Therefore, the screen contents can be inferred using supervised learning methods. Likewise, the screen content can be inferred via electromagnetic signals [58]. Nassi et al. [59] propose an speech inference attack, using the brightness of the power indicator LED to recover the acoustic signal generated by the target speaker. Similarly, Choi et al. [60] propose to use the electromagnetic signals generated by the power supply module on a Mixed-signal System-on-a-chip to achieve speech recovery. Zhu et al. [61] and Cheng et al. [62] propose to use CPU magnetic signals to infer the application launched on the target computer. The same attack can be achieved with power factor correction signals in the room's power supply lines [63].

Sensor Attacks. In addition to exploiting signals unintentionally generated by the device, physical signals can be injected into sensors on the devices to achieve sensor attacks. Trippel et al. [64] and Tu et al. [65] show the feasibility to inject signals accelerometers using ultrasound. Ji et al. [66] show that the signal injected into inertial sensors in camera anti-shake device can cause errors in image recognition systems. Zhang et al. and Yan et al. [67], [68] inject inaudible voice command into microphones using ultrasound exploiting the non-linearity of microphones. Furthermore, Ji et al. [69] exploit inverse piezoelectric effect of ceramic capacitors to generate the inaudible voice command through capacitors.

C. Enhancement Compared to `MagView`

The enhancement compared to `MagView` [70] is shown in Table VIII. First, the transmission rate is significantly improved from 8.9 bps to 59 bps, as we propose a new encoding scheme, i.e., the Timestamp-based scheme. Second, we evaluated the performance of both the Frame-type-and-QP-based scheme and the Timestamp-based scheme on various video players, and found that both schemes performed well when using software decoding. When using hardware decoding, the Frame-type-and-QP-based scheme failed but the Timestamp-based scheme could still transmit data on some video players. Third, we implemented and evaluated the forward error correction (FEC) on

TABLE VIII
ENHANCEMENT COMPARED TO MagView

Items	MagView	MagView++
Data embedding scheme	Frame-type-and-QP-based Scheme	Frame-type-and-QP-based Scheme & Timestamp-based Scheme
Transmission rate	Up to 8.9 bps	Up to 59 bps
Hardware decoding support	Not supported	Partially supported
Forward error code	Without	With

both schemes. The results show that the BER can be reduced to around 1/6 of the original, while the transmission rate is only reduced to 73.3% of the original.

XI. CONCLUSION

In this article, we propose a magnetic covert channel, which hides the data transmission in normal video playing tasks. Instead of controlling the CPU workload directly, MagView++ utilizes video as a medium to embed, transfer and finally leak the sensitive data via CPU magnetic field. Therefore, the sensitive data can be distributed with the video spread through the Intranet and played on any computer. MagView++ is stealthy as it hides CPU utilization changes in video decoding tasks, not influencing on the original video images. We design two data embedding schemes including a Frame-type-and-QP-based scheme used for smartphone receivers and a Timestamp-based scheme used for dedicated receivers to embed data into CPU utilization changes. We evaluate MagView++ under various settings including videos, device types, distances, background APPs, video players, and surroundings. When using smartphone receivers, MagView++ achieves up to 8.9 bps throughput with BER as low as 0.0057, while using dedicated receivers MagView++ achieves up to 59 bps throughput with BER as low as 0.0025. By using FEC, the BER of two schemes can further reduced to 1/6 of the original, at the cost of 23.7% loss of the throughput.

ACKNOWLEDGMENTS

Gang Qu's work was done when he visited Zhejiang University as a Qiushi Chair Professor.

REFERENCES

[1] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici, "AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in *Proc. Int. Conf. Malicious Unwanted Softw.*, 2014, pp. 58–67.
[2] E. Birk and J. Alexander, "Enabling smart phones in intel's factory," White Paper, Santa Clara, CA, USA: Intel IT, 2011.
[3] S. Biddle, "US military bans all physical media on its internal network," 2010, Accessed: Jul. 25, 2018. [Online]. Available: https://gizmodo.com/5711205/us-military-bans-all-physical-media-on-its-internal-network
[4] British Broadcasting Corporation, "IBM workers banned from using USB sticks," 2018, Accessed: Jul. 25, 2018. [Online]. Available: https://www.bbc.com/news/technology-44069488
[5] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, 1973.
[6] M. Guri and Y. Elovici, "Bridgeware: The air-gap malware," *Commun. ACM*, vol. 61, no. 4, pp. 74–82, 2018.
[7] Marie-Andree, "May an employer prohibit employees from taking pictures at the workplace?." 2014. Accessed: Jul. 30, 2019. [Online]. Available: http://www.maw-law.com/social-media-law/may-employer-prohibit-employees-taking-pictures-workplace/
[8] M. Guri, B. Zadov, and Y. Elovici, "ODINI: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1190–1203, 2019.
[9] M. Guri, "MAGNETO: Covert channel between air-gapped systems and nearby smartphones via CPU-generated magnetic fields," *Future Gener. Comput. Syst.*, vol. 115, pp. 115–125, 2021.
[10] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.
[11] ITU-T. H.265 high efficiency video coding, United Nations District, International Telecommunication Union, Geneva, 2018.
[12] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. Hoboken, NJ, USA: Wiley, 2004.
[13] M. Ghanbari, *Video Coding: An Introduction to Standard Codecs*, Michael Faraday House. Six Hills Way, Stevenage. Hem. SCI 2AY. United Kingdom: The Institution of Electrical Engineers. 1999.
[14] I. Hong, G. Qu, M. Potkonjak, and M. Srivastavas, "Synthesis techniques for low-power hard real-time systems on variable voltage processors," in *Proc. IEEE Real-Time Syst. Symp.*, 1998, pp. 178–187.
[15] K.-S. Lee, H. Wang, and H. Weatherspoon, "PHY covert channels: Can you see the idles?," in *Proc. 11th USENIX Symp. Networked Syst. Des. Implementation*, 2014, pp. 173–185.
[16] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser, "Covert channels using mobile device's magnetic field sensors," in *Proc. 21st Asia South Pacific Des. Automat. Conf.*, 2016, pp. 525–532.
[17] "x264," 2018, Accessed: May 16, 2018. [Online]. Available: https://www.videolan.org/developers/x264.html
[18] L. Merritt and R. Vanam, "Improved rate control and motion estimation for H.264 encoder," in *Proc. IEEE Int. Conf. Image Process.*, 2007, pp. 309–312.
[19] "FFmpeg: A complete, cross-platform solution to record, convert and stream audio and video," 2018, Accessed: May 16, 2018. [Online]. Available: https://www.ffmpeg.org/
[20] Wiebelhaus, "(52) Hikvision DS-2CD2085G1-I 8MP 2.8 mm darkfighter network bullet camera (color mode at night sample) - YouTube." 2019. Accessed: Jul. 30, 2019. [Online]. Available: https://www.youtube.com/watch?v=xFFaccsCfxY
[21] "psutil," 2018, Accessed: Jul. 31, 2018. [Online]. Available: https://github.com/giampaolo/psutil
[22] Keysight Technologies, "U2541A data acquisition," 2018, Accessed: Aug. 01, 2018. [Online]. Available: https://www.keysight.com/en/pd-1250127-pn-U2541A/250ksa-s-usb-modular-simultaneous-data-acquisition?cc=US&lc=eng
[23] Texas Instruments, "DRV425," 2018, Accessed: Aug. 1, 2018. [Online]. Available: http://www.ti.com/product/DRV425
[24] V. P. Kodali and V. Prasad, *Engineering Electromagnetic Compatibility: Principles, Measurements, Technologies, and Computer Models*. Hoboken, NJ, USA: Wiley-IEEE Press, 2001.
[25] Wikipedia Contributors, "Variable frame rate — Wikipedia, the free encyclopedia," 2021, Accessed: Jun. 3, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Variable_frame_rate&oldid=1008722054
[26] "Art technology USB3200 data acquisition card," 2021. [Online]. Available: http://c.gongkong.com/PhoneVersion/ProductDetail?pId=86419
[27] SIGMA CCTV, "Hikvision 60FPS 2MP IP camera video quality," 2021. [Online]. Available: https://www.youtube.com/watch?v=uYbQjqazfuY
[28] "MKVToolNix news – matroska tools for Linux/Unix and Windows," 2021. [Online]. Available: https://mkvtoolnix.download/
[29] Wikipedia, "Forward error correction — Wikipedia, the free encyclopedia," 2018, Accessed: Jul. 24, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Forward_error_correction&oldid=847049589
[30] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2005.
[31] Armstrong, Eur Ing Keith and Clough, Cherry, "EMC for systems and installations: Part 4," 2018, Accessed: Jul. 27, 2018. [Online]. Available: http://www.compliance-club.com/archive/keitharmstrong/systems_installations4.html

[32] N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross, "Control-flow bending: On the effectiveness of control-flow integrity," in *Proc. USENIX Secur. Symp.*, 2015, pp. 161–176.
[33] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch me, but don't touch me! Contactless control flow monitoring via electromagnetic emanations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1095–1108.
[34] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu, "On code execution tracking via power side-channel," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1019–1031.
[35] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "EDDIE: EM-based detection of deviations in program execution," in *Proc. 44th Annu. Int. Symp. Comput. Architecture*, 2017, pp. 333–346.
[36] Wikipedia, "Covert channel." 2019. Accessed: Jul. 31, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Covert_channel
[37] M. Guri, Y. A. Solewicz, A. Daidakulov, and Y. Elovici, "Acoustic data exfiltration from speakerless air-gapped computers via covert hard-drive noise ('diskfiltration')," in *Proc. 22nd Eur. Symp. Res. Comput. Secur.*, 2017, pp. 98–115.
[38] M. Guri, Y. Solewicz, and Y. Elovici, "Fansmitter: Acoustic data exfiltration from air-gapped computers via fans noise," *Comput. Secur.*, vol. 91, 2020, Art. no. 101721.
[39] M. Guri, "POWER-SUPPLaY: Leaking sensitive data from air-gapped, audio-gapped systems by turning the power supplies into speakers," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 313–330, Jan./Feb. 2023.
[40] P. Cronin, C. Gouert, D. Mouris, N. G. Tsoutsos, and C. Yang, "Covert data exfiltration using light and power channels," in *Proc. IEEE 37th Int. Conf. Comput. Des.*, 2019, pp. 301–304.
[41] M. Guri, B. Zadov, and Y. Elovici, "Led-it-go: Leaking (A lot of) data from air-gapped computers via the (small) hard drive LED," in *Proc. 14th Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2017, pp. 161–184.
[42] K. Zhang et al., "ChromaCode: A fully imperceptible screen-camera communication system," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 575–590.
[43] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "C3 APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage," in *Proc. 41st IEEE Symp. Secur. Privacy*, 2020, pp. 1728–1741.
[44] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "PowerHammer: Exfiltrating data from air-gapped computers through power lines," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1879–1890, 2020.
[45] S. K. Khatamifard, L. Wang, A. Das, S. Köse, and U. R. Karpuzcu, "POWERT channels: A novel class of covert communicationexploiting power management vulnerabilities," in *Proc. IEEE 25th Int. Symp. High Perform. Comput. Architecture*, 2019, pp. 291–303.
[46] R. Spolaor, L. Abudahi, V. Moonsamy, M. Conti, and R. Poovendran, "No free charge theorem: A covert channel via USB charging cable on mobile devices," in *Proc. Appl. Cryptogr. Netw. Secur.*, 2017, pp. 83–102.
[47] V. Coskun, K. Ok, and B. Ozdenizci, *Near Field Communication (NFC): From Theory to Practice*. New York, NY, USA: Wiley, 2011.
[48] H. Pan, Y. Chen, G. Xue, and X. Ji, "MagneComm: Magnetometer-based near-field communication," in *Proc. 23rd ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2017, pp. 167–179.
[49] B. Zhao, M. Ni, and P. Fan, "Powermitter: Data exfiltration from air-gapped computer through switching power supply," *China Commun.*, vol. 15, no. 2, pp. 170–189, Feb. 2018.
[50] N. Matyunin et al., "Tracking private browsing sessions using CPU-based covert channels," in *Proc. 11th ACM Conf. Secur. Privacy Wirel. Mobile Netw.*, 2018, pp. 63–74.
[51] M. Guri, M. Monitz, and Y. Elovici, "USBee: Air-gap covert-channel via electromagnetic emission from USB," in *Proc. IEEE 14th Annu. Conf. Privacy Secur. Trust*, 2016, pp. 264–268.
[52] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, "GSMem: Data exfiltration from air-gapped computers over GSM frequencies," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 849–864.
[53] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici, "BitWhisper: Covert signaling channel between air-gapped computers using thermal manipulations," in *Proc. IEEE 28th Comput. Secur. Found.s Symp.*, 2015, pp. 276–289.
[54] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *J. Cryptographic Eng.*, vol. 1, no. 1, pp. 5–27, 2011.
[55] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *Proc. 34th Annu. Cryptol. Conf. Adv. Cryptol.*, 2014, pp. 444–461.
[56] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 163–177.
[57] D. Genkin, M. Pattani, R. Schuster, and E. Tromer, "Synesthesia: Detecting screen content via remote acoustic side channels," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 853–869.
[58] N. Sehatbakhsh, B. B. Yilmaz, A. G. Zajic, and M. Prvulovic, "A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect.*, 2020, pp. 123–138.
[59] B. Nassi, Y. Pirutin, T. C. Galor, Y. Elovici, and B. Zadov, "Glowworm attack: Optical TEMPEST sound recovery via a device's power indicator LED," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 1900–1914.
[60] J. Choi, H. Yang, and D. Cho, "TEMPEST comeback: A realistic audio eavesdropping threat on mixed-signal SoCs," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1085–1101.
[61] Z. Zhu, H. Pan, Y. Chen, X. Ji, F. Zhang, and C. You, "MagAttack: Remote app sensing with your phone," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 241–244.
[62] Y. Cheng et al., "MagAttack: Guessing application launching and operation via smartphone," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2019, pp. 283–294.
[63] J. Zhang, X. Ji, Y. Chi, Y. Chen, B. Wang, and W. Xu, "OutletSpy: Cross-outlet application inference via power factor correction signal," in *Proc. ACM Conf. Secur. Privacy Wirel. Mobile Netw.*, 2021, pp. 181–191.
[64] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2017, pp. 3–18.
[65] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1545–1562.
[66] X. Ji et al., "Poltergeist: Acoustic adversarial machine learning against cameras and computer vision," in *Proc. IEEE 42nd Symp. Secur. Privacy*, 2021, pp. 160–175.
[67] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "DolphinAttack: Inaudible voice commands," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 103–117.
[68] C. Yan, G. Zhang, X. Ji, T. Zhang, T. Zhang, and W. Xu, "The feasibility of injecting inaudible voice commands to voice assistants," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1108–1124, May/Jun. 2021.
[69] X. Ji, J. Zhang, S. Jiang, J. Li, and W. Xu, "CapSpeaker: Injecting voices to microphones via capacitors," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 1915–1929.
[70] J. Zhang, X. Ji, W. Xu, Y. Chen, Y. Tang, and G. Qu, "MagView: A distributed magnetic covert channel via video encoding and decoding," in *Proc. IEEE 39th Conf. Comput. Commun.*, 2020, pp. 357–366.

Xiaoyu Ji (Member, IEEE) received the BS degree in electronic information and technology and instrumentation science from Zhejiang University, in 2010, and the PhD degree from the Department of Computer Science, Hong Kong University of Science and Technology, in 2015. He is now an associate professor with the Department of Electrical Engineering, Zhejiang University. His research interests include IoT security, including sensor, network, and AI security. He won the best paper award of ACM CCS 2017, ACM AsiaCCS 2018.

Juchuan Zhang (Member, IEEE) received the BS degree from the School of Electrical Engineering, Xi'an Jiao Tong University. He is currently working toward the PhD degree with USSLab, Zhejiang University. He is interested in physical side-channels. He did a series of work including exfiltrating data from airgapped networks via video encoding and decoding, inferring application launching through an outlet in another room, and injecting voices to microphones via capacitors.

Shan Zou (Member, IEEE) received the BS degree from the College of Electrical Engineering, Zhejiang University. He is currently working toward the graduate degree with USSLab, Zhejiang University. He is interested in side-channel attack, side-channel monitoring, and automatic driving security.

Gang Qu (Fellow, IEEE) received the BS and MS degrees in mathematics from the University of Science and Technology of China, in 1992 and 1994, respectively, and the PhD degree in computer science from the University of California, Los Angeles, in 2000. Upon graduation, he joined the University of Maryland, College Park, where he is currently a professor with the Department of Electrical and Computer Engineering and Institute for Systems Research. His primary research interests include the area of embedded systems and VLSI CAD with focus on low power system design and hardware related security and trust.

Yichao Chen (Member, IEEE) received the BS and MS degrees from the Department of Computer Science and Information Engineering, National Taiwan University, in 2004 and 2006, respectively,and the PhD degree in computer science from the University of Texas at Austin, in 2015. He joined Shanghai Jiao Tong University as a tenure-track assistant professor with the Department of Computer Science and Engineering, in 2018. His research interests include focus on networked systems and span the areas of wireless networking, network measurement, and analytics, and mobile computing.

Wenyuan Xu (Member, IEEE) received the BS degree in electrical engineering from Zhejiang University, in 1998, the MS degree in computer science and engineering from Zhejiang University, in 2001, and the PhD degree in electrical and computer engineering from Rutgers University, in 2007. He is a professor with the College of Electrical Engineering, Zhejiang University. Her research interests include wireless networking, network security, and IoT security. He received the NSF Career Award in 2009, a CCS best paper award in 2017, and an ASIACCS best paper award in 2018.